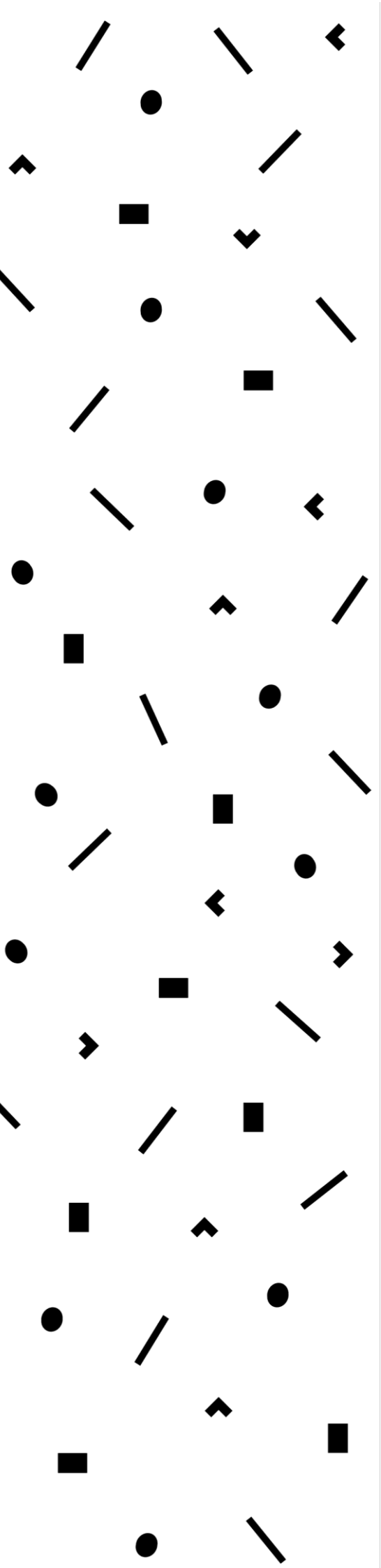


LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

# SHAREPOINT-SOVELLUKSET AZURESSA

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2015
Isto-Pekka Savolainen

Lahden ammattikorkeakoulu
Tietotekniikka

SAVOLAINEN, ISTO-PEKKA SharePoint-sovellukset Azuressa

Ohjelmistotekniikan opinnäytetyö 41 sivua

Kevät 2015

TIIVISTELMÄ

---

Opinnäyte toteutettiin kesällä 2014 JPP-Soft Oy:ssä. Työn tavoitteena oli siirtää custom-koodia pois SharePoint-ympäristöstä ja samalla saada useamman asiakkaan koodit helposti hallinnoitavaan paikkaan, tässä tapauksessa pilveen. Samalla selvitettiin, miten Azurea voidaan käyttää hyväksi SharePoint-sovellusalustana ja sovelluskehityksessä. Materiaalina käytettiin Microsoft developer network -palvelusta saatavaa dokumentaatiota sekä useita alan ammattilaisten blogikirjoituksia asiasta.

Pilviympäristönä käytettiin Microsoft Azurea isännöimään toteutettua remote provisioning -sovellusta. Toteutettu sovellus on osa isompaa tuotepakettia, ja sen ideana on luoda työtilasivustoja SharePointiin etänä pilvestä. Sovelluksessa noudatettiin CSOM-mallia sekä Provider-hosted apps -sovellusmallia. Opinnäytetyön aikana sovelluksesta tehtiin ensimmäinen versio, joka otettiin käyttöön muutamilla asiakkailla.

Asiasanat: Azure, SharePoint apps, CSOM, Provider-hosted apps

Lahti University of Applied Sciences
Degree Programme in Information Technology

SAVOLAINEN, ISTO-PEKKA: SharePoint apps in Azure

Bachelor's Thesis in software engineering, 41 pages

Spring 2015

ABSTRACT

---

This Bachelor's Thesis was carried out at JPP-Soft Oy during summer 2014. The objective of the study was to move custom code out of SharePoint environments to a place where all of the customer's code would be more manageable, which in this case is cloud. At the same time Azure, and how it can be used with SharePoint app development and as an app hosting service, was studied. Documentation provided by the Microsoft development network and various blogs from professionals was used as research material during the study.

Microsoft Azure was used to host the created remote provisioning application as a cloud service. The created application was a part of a larger product, and its purpose was to create workspace-sites remotely from the cloud environment. The application logic was coded using the CSOM model, and the application is a Provider-hosted app. The first version of the application was installed on a few customer environments.

Key words: Azure, SharePoint apps, CSOM, Provider-hosted apps

SISÄLLYS

# 1 JOHDANTO

SharePoint on kokenut viime vuosina valtavia muutoksia ja yhä useampi yritys on siirtynyt SharePoint-pohjaisiin ratkaisuihin sekä intranetinä että ulkoisina verkkosivuina. Ratkaisut eivät enää ole liian kalliita ja monimutkaisia hallittavaksi osittain pilvessä toimivan Office 365:n ansiosta. Versiossa 2013 tullut uusi sovellusmalli tarjoaa ratkaisumalleja ja mahdollisuuksia kehittää entistä tehokkaampia ja helpommin hallittavia sovelluksia.

Microsoft on toivonut, että custom-koodia vähennettäisiin SharePoint-ympäristöissä. Yksi tapa vähentää koodia on siirtää sovellukset pilveen, jossa niitä voidaan ajaa erillään SharePointista. Projektin yhtenä tavoitteena on siirtää osa asiakkaiden sovelluksista pilveen, tässä tapauksessa Azureen, josta niitä olisi helpompi hallita.

Opinnäytetyö pohjautuu JPP-Soft Oy:llä kesällä 2014 toteutettuun sivustonluonti-sovellukseen. Sovelluksella pystytään luomaan räätälöityjä työtila-sivustopohjia hyväksi käyttäen alisivustoja haluttuihin sivustokokoelmiin etänä Microsoft Azuresta. Toteutettu sovellus on niin sanottu Provider-hosted-sovellus, ja on osa laajempaa tuotepakettia. Tuotepaketti on käytössä useilla asiakkailla ja sovellus olikin siksi saatava mahdollisimman nopeasti käyttökuntoon.

Työssä tutustutaan yleisesti SharePoint-sovelluskehitykseen ja pohditaan, miten Azurea voidaan hyödyntää yleisestä näkökulmasta sovelluksissa ja sovelluskehityksessä. Mitä hyötyjä SharePointin sovellusten asentamisesta Azureen on?

Toteutettu ohjelma on kirjoitettu C#-kielellä noudattaen SharePoint client object -mallia (CSOM). Tutkimuslähteinä käytettiin Microsoftin ohjelmistokehittäjien dokumentaatiota ja mallikoodeja Azuresta sekä SharePointista. MSDN:n lisäksi hyödynnettiin useita ammattilaisten SharePoint-aiheisia blogikirjoituksia Microsoftin virallisilta kanavilta, kuten Technetistä, sekä muilta erillisiltä blogisivustoilta.

Testiympäristönä käytettiin Azure-verkkosivustoa sekä Office 365 -kehitysympäristöä. Ohjelmoimiseen käytettiin Visual Studio 2013:a Office 365 -

lisäosalla sekä versionhallintana Microsoftin Team Foundation -palvelinta. Testaukseen käytettiin selaimessa toimivaa TestRail-palvelua, johon kirjattiin ylös manuaalisesti suoritettavia testitapauksia.

# 2 SHAREPOINT JA AZURE YLEISESTI

## 2.1 SharePoint

SharePoint on Microsoftin tekemä selaimessa käytettävä tietojen tallennus-, järjestely- ja jakamispaikka. SharePointia voidaa käyttää muunmuassa intranetinä, wikinä, verkkosivustona, projektinhallintajärjestelmänä ja BI-järjestelmänä. Järjestelmä on suunniteltu pääosin keski- ja suuren koon organisaatioille, mutta sitä voidaan käyttää pienemmissäkin organisaatioissakin.

SharePointin ensimmäinen versio julkaistiin vuonna 2001. Tämän jälkeen on julkaistu versiot 2003, 2007, 2010 ja 2013 sekä Office 365 -pakettiin kuuluva SharePoint Online. SharePoint 2016 -version julkaisusta on tietoa, mutta ei vielä varmuutta julkaisupäivästä.

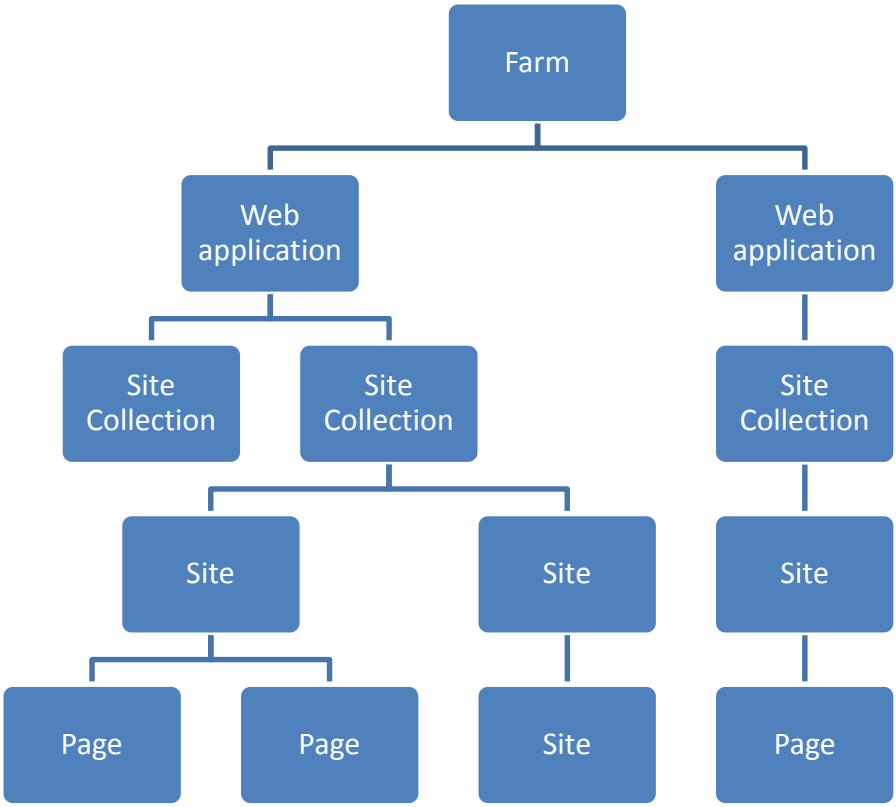
SharePoint on käynyt läpi useita suuria muutoksia, joista tähän mennessä isoimmat ovat olleet versioissa 2010 ja 2013. Nykyään SharePointista saa paikallisen version, pilviversion sekä paikallisen- ja pilviversion hybridin. Paikallinen versio, eli SharePoint 2013, on enemmän muokattavissa ja tukee muun muassa farmitason koodausta. Vaikka SharePoint 2013:a isännöidäänkin yleensä paikallisella palvelimella, voidaan sitä isännöidä myös virtuaalikoneella pilvessä, esimerkiksi Azuressa. Paikallisesta SharePoint Server 2013:sta on kolme päätuotetta, Foundation, Standard sekä Enterprise, jotka soveltuvat erikokoisille organisaatioille ja tarpeille. (MSDN 2013)

Pilvessä käyttöön saa huomattavasti vähemmän ominaisuuksia, joihin voi itse vaikuttaa, mutta ylläpito on vähemmän vaativaa, koska pilvessä sijaitsevaa palvelinta tarvitsee hallinnoida vähemmän. Hybridi-versiota voidaan käyttää esimerkiksi intranetissä, josta osa sivuista, kuten projektisivut, halutaan jaettavaksi yrityksen ulkopuolisille tahoille niin sanottuna extranetinä. Tällöin sivuston intranet-osuus sijaitsisi omalla palvelimella paikallisesti ja käyttäisi SharePoint server 2013:a ja projektisivut olisivat osa esimerkiksi O365:ssa sijaitsevaa pilvipalvelinta SharePoint Onlinessa. Hybridi-versiossa on paljon muuntelemisen varaa, muun muassa hakua pystytään konfiguroimaan joko yksi- tai kaksisuuntaiseksi pilven ja paikallisen toteutuksen välillä. (Technet 2014)

## 2.2 SharePoint sivustotasot

SharePointin topologiassa jokainen sivustokokoelma on jonkin web-applikaation alla ja Farmi pitää allaan jokaista web-applikaatiota. Jokainen sivusto sijaitsee sivustokokoelmassa. Kuten kuviosta 1 näkyy, sivustokokoelmassa voi olla useita sivustoja ja näillä sivustoilla voi olla useita alisivustoja. Sivustokokoelmille voidaan määrittää oma tyyli, joka voidaan periyttää jokaiselle alisivustolle. Kokoelmia voidaan käyttää hyväksi määriteltäessä sivustojen hallintarajoja. Tarvittaessa voidaan määritellä sivustokokoelman administraattoreita sekä sivustojen administraattoreita, joilla ei ole oikeuksia hallinnoida sivustokokoelmaa, tai muita sivustokokoelman alisivustoja. Listat, verkko-osat ja sovellukset ovat yleensä sivustokokoelma kohtaisia, mutta niitä voidaan tarvittaessa käyttää muiltakin sivustokokelmilta. (Technet 2014)

Farmeissa on yleensä useita sivustokokoelmia, koska kokoelmia käytetään sivustojen lisäksi isännöimään SharePointille tärkeitä palveluita. Tällaisia palveluita on muun muassa hakukeskus ja käyttäjien henkilökohtaiset profiilit sisältävä My Sites. (Technet 2014)




KUVIO 1. SharePoint-topologia

## 2.3 Sivustopohjat yleisesti

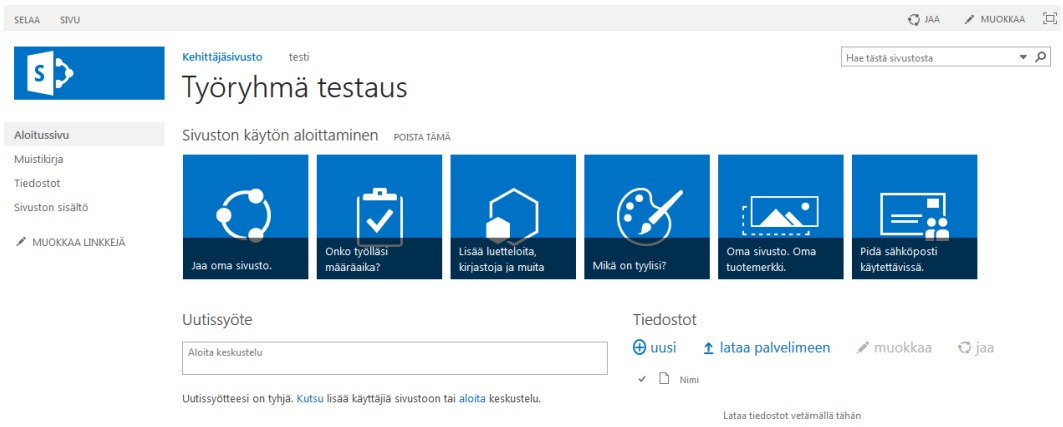
Sivustot luodaan aina jollakin templatella, joka määrittää sivujen tyylin ja ominaisuudet. SharePoint tarjoaa useita Out of Box (OOB) -sivustopohjia valmiiksi tuotteissaan. Yleisiä sivustojen templateja ovat tiimi-, blogi-, projekti-, yhteisö-, dokumenttikeskus-, tallennekeskus-, BI-, hakukeskus- sekä julkaisu-sivusto. Sivustopohjat jaetaan kolmeen kategoriaan: yhteistyö, yritys sekä julkaisupohjat. Nämä valmispohjat ovat SharePointissa käytössä riippuen ostetusta tuotteesta. Taulukosta yksi näkee sivustopohjien saatavuuden. Myös itse tehtyjä custom-sivustopohjia voidaan käyttää sivustokokoelmien ja sivustojen luomiseen. (Office 2015)

TAULUKKO 1. OOB-sivustopohjat (Office 2015)

| | | | | O365 for business | | SharePoint | |
|---|---|---|---|---|---|---|---|
| **Kategoria** | **Sivutyyppi** | **Sivusto-kokoelma** | **Sivusto** | **Small** | **Medium and Enterprise** | **Server 2013** | **Foundation 2013** |
| Yhteistyö | Tiimi | Kyllä | Kyllä | Kyllä | Kyllä | Kyllä | Kyllä |
| Yhteistyö | Blogi | Kyllä | Kyllä | Kyllä | Kyllä | Kyllä | Kyllä |
| Yhteistyö | Projekti | Kyllä | Kyllä | Kyllä | Kyllä | Kyllä | Ei |
| Yhteistyö | Yhteisö | Kyllä | Kyllä | Ei | Kyllä | Kyllä | Ei |
| Yritys | Dokumentti-keskus | Kyllä | Kyllä | Ei | Kyllä | Kyllä | Ei |
| Yritys | Tallenne-keskus | Kyllä | Kyllä | Ei | Kyllä | Kyllä | Ei |
| Yritys | BI-keskus | Kyllä | Kyllä | Ei | Kyllä | Kyllä | Ei |
| Yritys | Hakukeskus | Kyllä | Kyllä | Ei | Kyllä | Kyllä | Ei |
| Julkaisu | Julkaisu | Ei | Kyllä | Ei | Ei | Kyllä | Ei |
| Julkaisu | Julkaisu työnkululla | Ei | Kyllä | Ei | Ei | Kyllä | Ei |
| Julkaisu | Yrityswiki | Kyllä | Kyllä | Ei | Kyllä | Kyllä | Ei |

## 2.4 Valmiit sivustopohjat

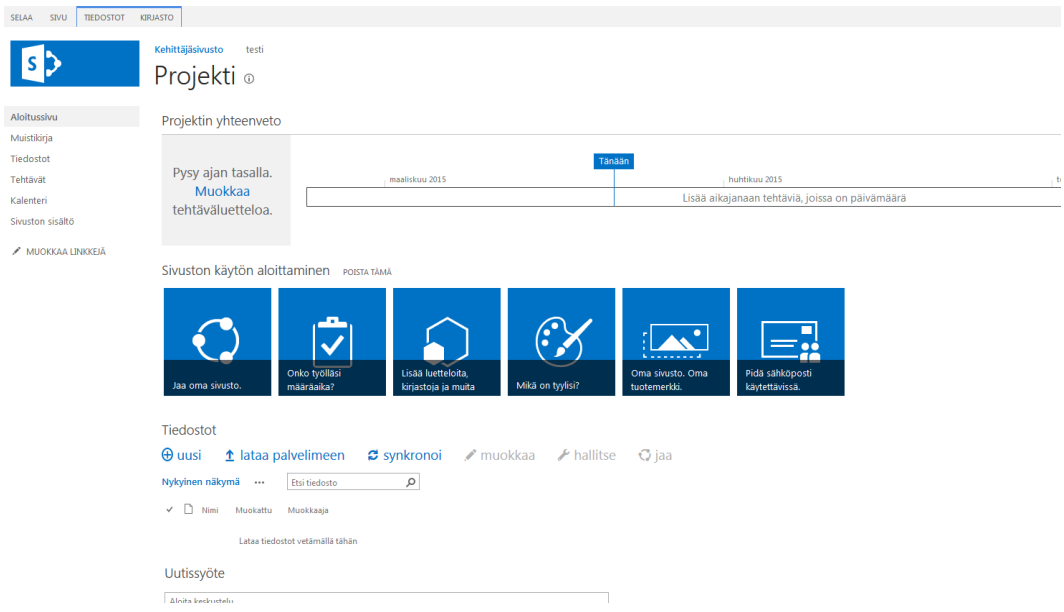
Tässä luvussa esitellään muutamia OOB-sivustopohjia ja niiden ominaisuuksia. Oletuksena työryhmäsivustolla on käytössä kirjastot ja listat jaetuille dokumenteille, ilmoituksille, kalenterille, linkeille, tehtäville ja keskustelualueelle. Kuviossa 2 on esitelty valmis työryhmäsivusto oletustyyleillä.




KUVIO 2. OOB-työryhmäsivusto

Julkaisusivustoa käytetään usein sisältösivuna esimerkiksi intranetissä tai julkisilla verkkosivuilla. Sivustolla on oletuksena käytössä sivujen versiointi sekä hyväksyntä -ominaisuudet, joiden avulla tekstin ja julkaisuiden muokkaaminen helpottuu.

Kuten kuviosta 3 voidaan todeta, projektisivu on pitkälti samanlainen ominaisuuksiltaan kuin tiimisivusto. Sivustolla on aikajana sekä tehtäväluettelo, jotka seuraavat sivustolle lisättyjä tehtäviä.




KUVIO 3. OOB-projektisivusto

## 2.5 Custom-sivustopohjat

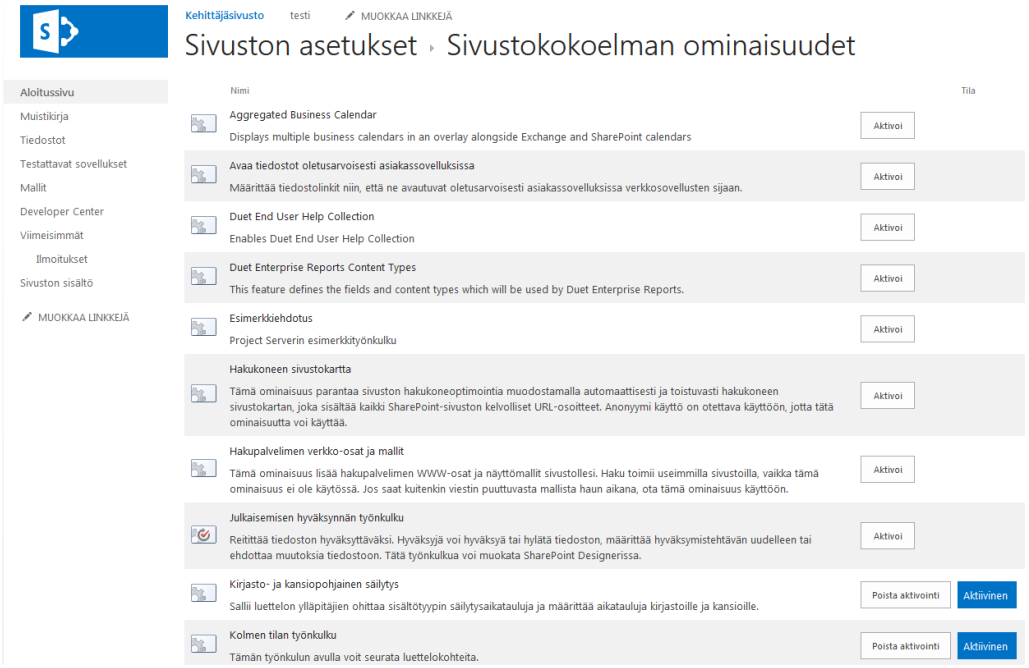
Sivustoja voidaan luoda customoiduista sivustopohjamalleista. Mallipohjien kehitys ja muokkaus tehdään Visual Studio -työkalulla ja niitä voidaan ladata SharePointin ratkaisukatalogiin, josta ne ovat käytössä sivustokokoelmassa. Jo asennetusta sivustosta voidaan ottaa kopio käytettäväksi sivustopohjana. Pohjien avulla voidaan sivustonluonnin yhteydessä asettaa paikalleen muun muassa sivuja, sisältötyyppejä, master-tiedostoja, moduuleita, näkymiä, kirjastoja, työnkulkuja, listoja sekä sisältöä listoihin ja kirjastoihin (Office 2015). Sivustot noudattavat sivustokokoelman asetuksia ominaisuuksien käytössä, joten jos jotain ominaisuutta ei ole tuettu, ei sivustopohjaa voida käyttää, ennen kuin ominaisuus otetaan käyttöön sivustokokoelmassa.

Custom-sivustopohjia tehdään sivustoluonnin sulavoittamiseksi ja helpottamiseksi. Usein SharePointia kehittävät ohjelmistotalot myyvät räätälöityjä sivustopohjia tuotteissaan.

## 2.6 Sivusto- ja sivustokokoelman ominaisuudet

SharePointissa on kokonaisuudessaan neljän eri tason aktivoitavia ominaisuuksia: palvelintaso, verkkosovellustaso, sivustokokoelmataso sekä sivustotaso, joista kaksi jälkimmäistä ovat aktivoitavissa sivuston asetukset -välilehdellä perus -sivustoilla. SharePointin ominaisuusmalli ei ole muuttunut huomattavasti 2010 version jälkeen. Tässä luvussa keskitytään sivustokokoelmatason sekä sivustotason ominaisuuksiin. Kuviossa 4 on joitakin sivustokokoelmatasolla saatavilla olevia ominaisuuksia, joita voidaan halutessa kytkeä päälle tai pois.

Ominaisuuksia voidaan luoda ja liittää sivustoille paketteina SharePoint -ratkaisuissa. Ominaisuudet ovat käytännössä manifest XML -tiedostoja, joissa on lista eri avaruuksissa (scopes) sijaitsevista tiedostoista, joita siirretään sivustokokoelmaan aktivoitaessa. XML-tiedosto voi pitää sisällään muun muassa pohjia, sivuja, listoja, tapahtumankäsittelijöitä, työnkulkuja ja muita custimisoituja komponentteja. Ominaisuuspaketteja voidaan avata, muokata ja paketoida Visual Studiolla.




KUVIO 4. OOB sivustokokoelmatason ominaisuuksia

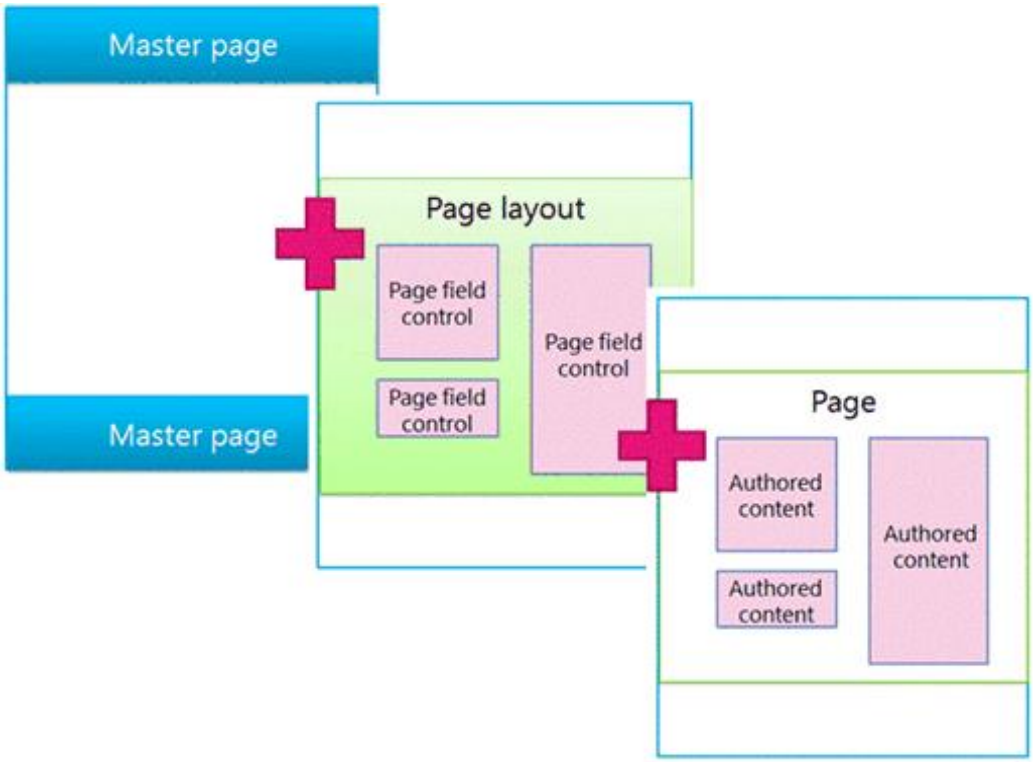
Ominaisuuspaketit saattavat muistuttaa malliltaan sovelluksia, mutta ne eroavat seuraavilla tavoilla: Ominaisuuksilla lisätyt komponentit ovat osa SharePoint-sivustoa, mutta sovelluksilla listätyt komponentit sijaitsevat omassa sovellusverkossa. Ominaisuuksilla lisätty custom-koodi ajetaan SharePoint-serverillä, toisin kuin sovelluksissa, joissa koodi ajetaan esimerkiksi asiakkaan koneella tai pilvessä. Komponenteilla on suora pääsy koko sivuston muihin komponentteihin, toisin kuin sovelluksien komponenteilla.

## 2.7 Sivuston brändäys

SharePoint-sivustojen brändäys eroaa perinteisestä webbisuunnittelusta monellakin tapaa. Pelkkä HTML, CSS ja Photoshop osaaminen ei riitä, ja suunnittelijoilta vaaditaan myös laajaa SharePointin tuntemusta.

Sivustoilla on käytössä Master-sivu, joka toimii jokaisen sivustokokoelman sivun kehikkona. Master-sivulla määritellään sivujen logo, navigaatio, menu-valikko sekä alatunniste. Jokainen erillinen sivu käyttää asp-pohjaisia sivumalleja, joissa sijaitsevat kontrollit muun muassa sivuston sisällölle, otsikoille, verkko-osa-

alueet. Sivumallin päälle liitetään vielä itse sivu, johon kaikki sisältö, kuten teksti tai verkko-osat, voidaan siirtää. Kuviosta 5 voidaan nähdä, miten sivuja piirrettäessä master-sivu, sivumalli ja sivu yhdistetään. Muokatut sivupohjat, master-sivut, JavaScript-tiedostot ja CSS-tiedostot ladataan sivustokokoelman Master Page -galleriaan, josta ne ovat käytössä kaikilla alisivuilla.(MSDN 2013).




KUVIO 5. Sivutasot (MSDN)

## 2.8 Microsoft Azure

Microsoft Azure on pilvipalvelu infrastruktuuri (IaaS) ja alusta (PaaS) erinäisille sovelluksille ja palveluille. Azure tukee useita Microsoftin ja kolmansien osapuolien sovelluksia, systeemeitä, kehyksiä ja ohjelmointikieliä. Azure tukee muun muassa ASP.NET-, PHP-, Node.js- ja Python-ohjelmointikieliä sivustojen rakentamisessa sekä FTP:tä, Gitiä, Mercurialia ja TFS:ää niiden levittämisessä. Sinne voidaan siirtää valmiita infrastruktuureita ja ohjelmistoja ajettavaksi Windows- ja Linux-virtuaalikoneissa. Ohjelmistoja voidaan myös kehittää

käyttämällä Azuren virtuaalikoneita pilvipalvelualustana. Azure toimii tietokantana esimerkiksi aktiivihakemistona tai SQL-datakeskuksena. Kuviossa kuusi on listattuna suurin osa Microsoft Azuren tarjoamista palveluista. (Microsoft 2014)


KUVIO 6. Azure palvelut (Microsoft 2014)

## 2.9 SharePoint-ratkaisut Azuressa

SharePoint-farmeja ja palveluita voidaan asentaa Azureen ajettavaksi erikokoisilla virtuaalikoneilla. Paikallisesta SharePoint-ympäristöstä voidaan tehdä hybridimalli vaihtoehtoisesti O365:n sijaan yhdistämällä joitakin paikallisia palveluita Azureen. Aktiivihakemisto- ja DNS-palvelut ja ulospäin näkyvät sivustot voidaan isännöidä Azuressa. Hybridimalli voidaan toteuttaa yhdistämällä SharePoint ja Azure VPN-yhdyskäytävällä. Tämänlaisessa ratkaisussa säästetään muun muassa laitteistokustannuksissa.

Microsoft suosittelee Azuren käyttöä SharePoint-sovellusten kehityksessä ja testauksessa, koska näitä ympäristöjä on helppo luoda ja hallinnoida. SharePoint kehitysympäristö voidaan luoda virtuaalikoneeseen Azuressa. Kehitys -ympäristöissä on valmiiksi asennettuna tarvittavat työkalut sovelluskehitykseen. Virtuaalikoneita voidaan käynnistää, sammuttaa ja luoda tarvittaessa ja nopeasti.

Azure tarjoaa useita valmiusmalleja onnettomuustilanteita varten. Valmiustilaa voidaan käyttää farmin varadatakeskuksena toisen SharePoint-serverin sijaan, jolloin onnettomuuden sattuessa farmin palauttaminen olisi halvempaa ja tehokkaampaa. (Technet 2014.)

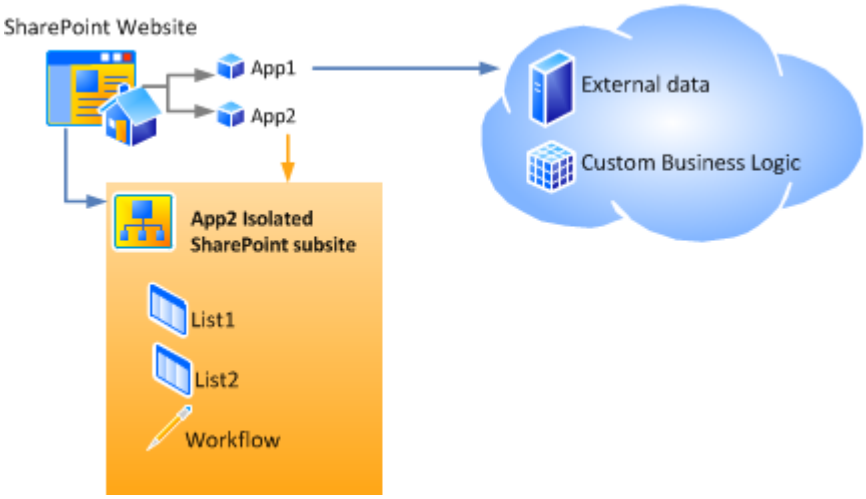
Sovelluksia voidaan isännöidä, testata ja rakentaa Azuressa, jolloin SharePoint-ympäristön voi jättää lähes customkoodivapaaksi.

# 3 SHAREPOINT SOVELLUSKEHITYS

Tässä luvussa keskitytään pilvessä toimivaan SharePoint Onlineen, sekä paikallisesti toimivan SharePoint 2013 -sovelluskehitykseen. Lisäksi tutustutuaan lyhyesti myös SharePoint 2013 -sovelluskehityksen ja aikaisempien versioiden eroihin.

Nyky SharePointissa sovellukset eivät ole suoraan osa SharePoint-sivustoja, vaikka ne asennetaankin sivustokokoelmaan, vaan ne ovat sivustoista irrallisia sovelluksia, jotka toimivat omassa sovellusverkossa. Kuviossa 7 on esimerkki SharePoint-isännöidystä sekä erillisellä palvelimella sijaitsevasta sovelluksesta. Sovellukset ajetaan asiakkaan koneella tai erillisellä palvelimella, ja koodin ajaminen asiakkaan koneella vaatii .NET-kehyksen. Sovelluksille myönnetään asennettaessa oikeudet sivuille käyttämällä OAuth-teknologiaa.

Sovellukset voivat sisältää muun muassa listoja, työnkulkuja, näkymiä, sisältötyyppejä ja muita SharePoint-komponentteja. Myös komponentit asennetaan samaan sovellusverkkoon.

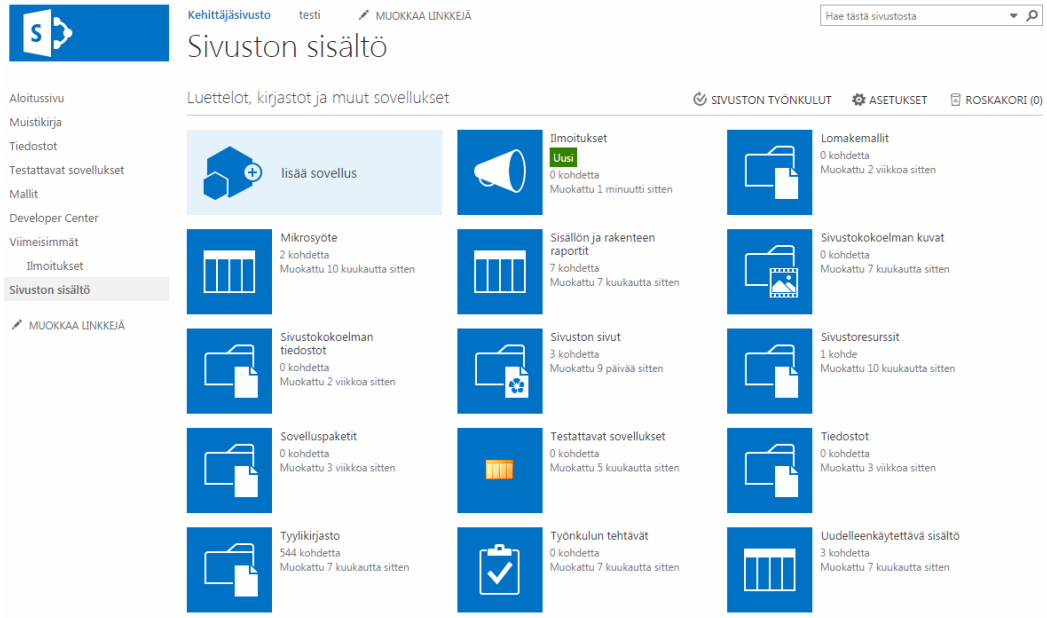



KUVIO 7. Sovellus pilvessä ja sovellusverkossa (MSDN 2015)

SharePoint 2010 ja aikaisemmissa versioissa ratkaisujen koodit ajettiin kokonaan SharePointin serverillä, farmi-tasolla. Sovellukset ovat lisäpalveluita, jotka ajetaan erillään SharePointista. SharePoint 2013:a edeltävissä versioissa ei ole ollut

vastaavaa sovellusmallia, joten kustomoitu sisältö asennettiin sivustoille erilailla. Ennen nykyistä sovellusmallia versiossa 2010 käytettiin NCSS:ää (no-code sandboxed solutions), eli kooditonta hiekkalaatikkomallia. NCSS-mallissa oli lähes kaikki samat ominaisuudet, kuin sovellusmallissa, mutta käytössä ei ollut custom hallittua koodia (managed code). Hallittu koodi on Microsoftin termi .NET-koodille, joka ajetaan Common Language Runtimessa (CLR). Nykyään Microsoft toivoo, että NCSS-mallista siirryttäisiin sovellusmalliin (MSDN 2014).

Sovelluksia voidaan lisätä sivuille sivuston sisältö välilehdeltä, josta pääsee käsiksi sovelluskauppaan, sovelluskatalogissa jaettaviin sovelluksiin sekä OOB sovelluksiin. Kuviossa 8 on sivustolle asennettu muutamia OOB-sovelluksia.




KUVIO 8. Sivuston sisältö

## 3.1 Mitä SharePointin sovelluksilla voidaan tehdä

Iso osa räätälöidystä sisällöstä voidaan julkaista SharePointiin sovelluksena. Sovelluksilla voidaan hallita lähes kaikkia SharePointin merkittävimmistä komponenteista, jotka asentuvat silloin samaan verkkoon sovelluksen kanssa. Tärkeita komponentteja ovat muun muassa seuraavat:

- ominaisuudet (web-scope)
- custom-toimenpiteet, kuten pikakuvakkeet ja komentonauhan customoinnit
- etätapahtumankäsittelijät
- verkko-osien ja sovellusosien koodi
- CSS- ja JavaScript-tiedostojen käyttö SharePoint-sivuilla
- moduulit
- sivut
- listamallit
- listat ja kirjastot
- lomakkeet
- näkymät
- sisältötyypit
- kenttätyypit
- työnkulut
- sivupohjat.

Sovelluksia voidaan liittää SharePointin käyttöliittymään monella tapaa. Sovellus voi olla esimerkiksi valintanauhasta käynnistettävä ohjelma tai verkko-osaa muistuttava sovellusosa sivustolla, kuten lista tai lomake (MSDN 2014). Sovellus voi olla myös SharePoint-sivua muistuttava kokosivun sovellus, joka on perinyt isäntäsivustonsa tyylin.

Sovelluksien avulla SharePointtiin voidaan integroida useita kolmannen osapuolen palveluita, kuten Twitter. Kolmannenosapuolen palvelut liitetään sivuille yleensä sovellusosina (App parts). Sovellusosat ovat uusi ominaisuus SharePoint 2013:ssa, joiden on tarkoitus korvata aikaisemmin käytössä olleet

verkko-osat (Web parts). Verkko-osat ovat kevyempi vaihtoehto sovellusosiin, ja yritykset vielä tälläkin hetkellä rakentavat custom-verkko-osia SharePoint 2013 -sivustoille. Sovellus -ja verkko-osien erot ovat peruskäyttäjälle lähes olemattomat. Osien ero nähdään niiden kehityksessä. Uudet sovellusosat on suunniteltu ajettavaksi SharePointin ulkopuolella sovellusverkossa, josta ne liitetään sivustoille IFRAME-teknologian avulla. Vanhat verkko-osat ajetaan suoraan SharePointin serverissä ja ovat osa verkkosivua. Uuden sovellusmallin ansiosta huonosti tehdyt sovellusosat eivät haittaa verkkosivua, toisin kuin huonosti tehdyt verkko-osat.

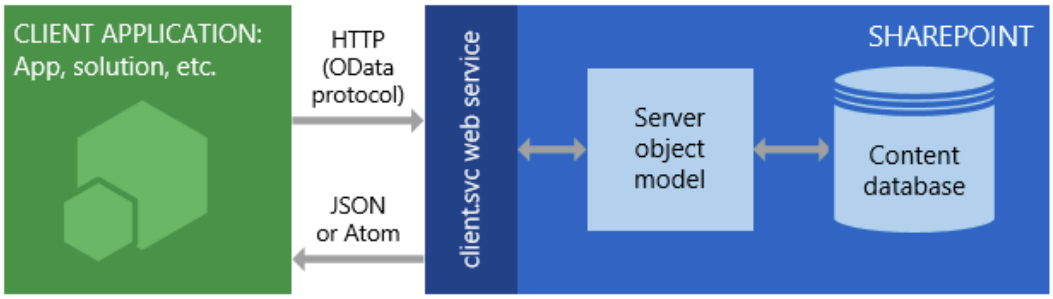
Oikeustaso myönnetään vain kerran sovellukselle, minkä ansiosta niillä voi olla enemmän oikeuksia sivustolle, kuin sovellusta käyttävällä asiakkaalla on. Sovelluksia ei kuitenkaan voida asentaa sivustokokoelmaan, jos asentajalla ei ole täysiä oikeuksia kyseisessä kokoelmassa.

## 3.2 REST ja CSOM

SharePointista voidaan hakea dataa Representational State Transfer -arkkitehtuurilla (REST) tai Client Object Model-mallilla (CSOM). CSOM-malli on suunniteltu korvaamaan Server-side object -malli (SSOM), sillä CSOM -mallilla voidaan tehdä lähes kaikki samat toimenpiteet sivustokokoelmatasolla, kuin SSOM-mallilla. Sekä REST- että CSOM-mallilla voidaan tehdä lähes samoja toimenpiteitä, joten se on yleensä kehittäjästä eikä vaatimuksista kiinni, kumpaa käytetään sovelluskehityksessä. REST-palvelua käytetään paljon SharePointin ulkopuolisessa ohjelmistokehityksessä, joten se on usein jo entuudestaan tuttu uusille SharePoint-kehittäjille. CSOM ja sitä edeltävä SSOM ovat olleet jo pitkään tärkeä osa ohjelmistokehitystä, joten usein kokeneemmat ohjelmistokehittäjät valitsevat sen käytettäväksi työkaluksi. Opinnäytetyön yhteydessä tehdyssä sovelluksessa käytettiin CSOM-mallia lähinnä sen oppimisen takia. (MSDN 2015)

REST on palvelu, jonka avulla sovellusverkosta voidaan tehdä CRUD-operaatioita (create, read, update, delete) SharePointin eri komponentteihin. REST perustuu URL-osoitteen avulla tehtyihin hakuihin, jotka lähetetään HTTP:llä

käyttäen OData-protokollaa SharePointin REST-palveluun, minkä jälkeen palvelu lähettää pyydetyn datan JSON- tai Atom XML -muodossa takaisin. Kuviossa 9 esitellään RESTin toimintamalli. (MSDN 2015)



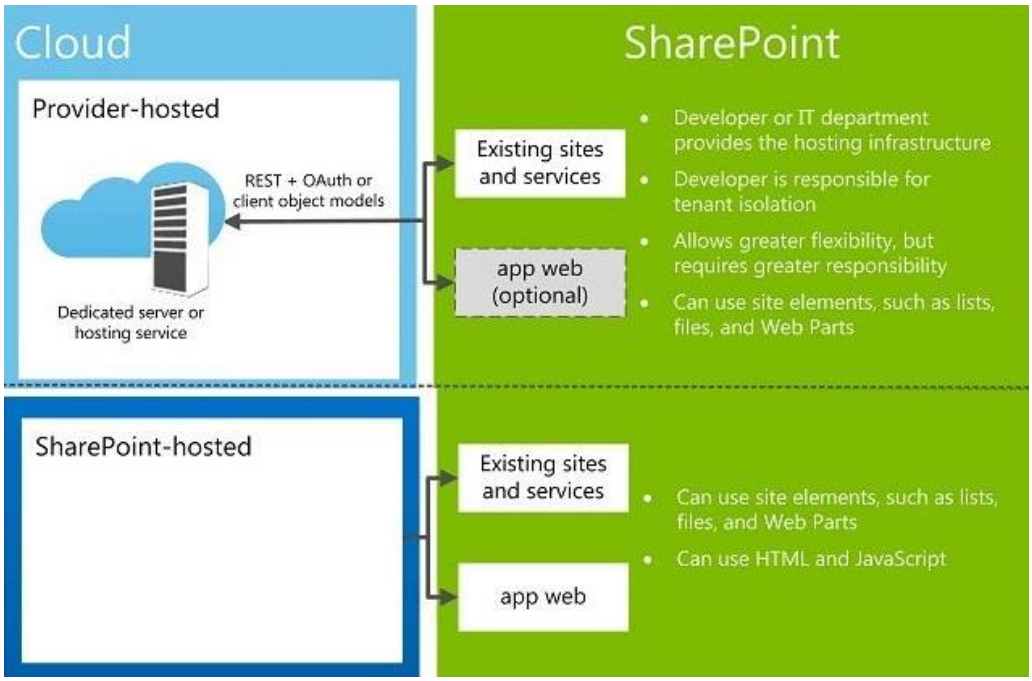

KUVIO 9. REST-toimintamalli

Kuten edellä mainittiin, CSOM-malli perustuu .NET-kehykseen ja SharePoint-palvelimella ajettavaan SSOM-malliin. CSOM:n avulla voidaan tehdä samoja toimenpiteitä, kuin aikaisemmin esitellyllä REST-mallilla. Näiden lisäksi CSOM:llä pystytään muokkaamaan hallittuja metadata -luokituksia sekä muokkaamaan työnkulkuja. Kyselyt perustuvat Microsoft.SharePoint.Client-kirjastoon, jonka avulla sivustoihin otetaan yhteys lataamalla sivuston konteksti URL-osoitteen perusteella. Muokkauskyselyitä listoihin ja muihin komponentteihin muodostetaan CAML-kielellä. CAML on XML pohjautuva kieli, jota käytetään useiden SharePoint-komponenttien kuvauskielenä. (MSDN 2011)

## 3.3 Sovellusten isännöinti

SharePoint-sovellus voidaan isännöidä paikallisesti lataamalla se sovelluskatalogista tai sovelluskaupasta. Sovellus sijaitsee tällöin eristetyssä aliverkossa nimeltä sovellusverkko, ja sen koodi ajetaan asiakkaan koneella. Paikallisesti isännöidyissä sovelluksissa ei sallita palvelinpuolen koodia. Kaikki sovelluksen omat komponentit ovat asennettuina sovellusverkkoon, ja ainoastaan JavaScriptillä voidaan kysellä dataa farmitasolta. (MSDN 2015)

Vaihtoehtoisesti sovelluksia voidaan isännöidä etänä jossakin pilvipalvelussa, kuten Microsoft Azuressa. Menetelmää kutsutaan nimellä Provider-hosted apps. Kuviossa 10 nähdään tämänhetkiset sovellusten isännöintivaihtoehdot.

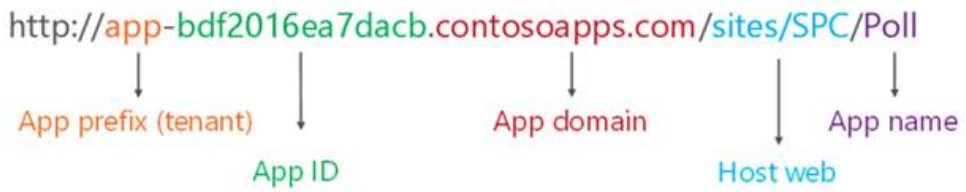



KUVIO 10. Sovellusten isännöinti (MSDN 2015)

Aina, kun asennetaan uusi SharePointissa isännöitävä sovellus, sille luodaan oma sovellusverkko asennetussa sivustokokoelmassa. Vaikka farmissa olisi samaa sovellusta useammassa sivustokokoelmassa, jokaisella olisi silti oma uniikki sovellusverkko. Tämä edesauttaa yleistä tietoturvaa SharePointissa, sillä omasta sovelluksesta ei ole pääsyä toisen sovellusverkon dataan helposti.

SharePoint-farmissa täytyy konfiguroida sovellusalue, jotta sovelluksia voidaan asentaa. Sovellusalueen lisäksi farmissa kytketään päälle palveluita, jotka muun muassa generoivat ja pitävät kirjaa sovellustunnuksista (App ID). Koska sovellukset sijaitsevat omassa erillisessä sovellusverkossa muusta SharePointista, eroaa niiden URL-osoite muista sivustoista. Sovellusverkon osoite muodostuu useammasta osasta, johon kuuluvat sovellusetuliite (App prefix), uniikki sovellustunnus (App ID), sovellusisäntä (App domain) sekä sovelluksen sijainti

(Host web) ja nimi (App name). Kuviossa 11 on esimerkki mahdollisesta sovelluksen osoitteesta (Whitehead 2013).




KUVIO 11. Sovelluksen osoite (Whitehead 2013)

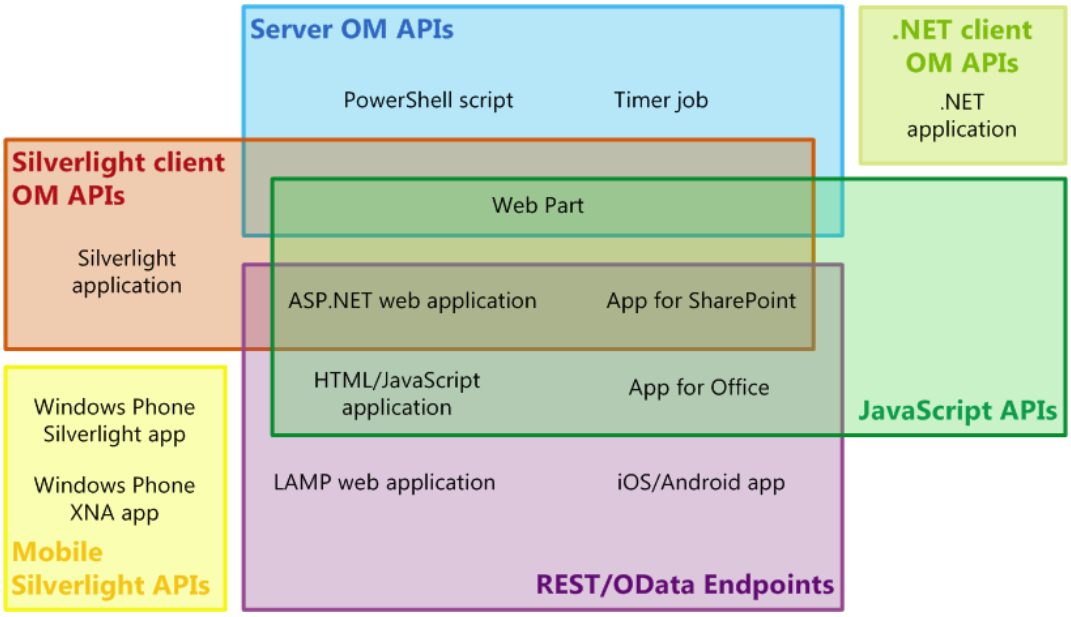
Jokaisella sovelluksella on oma uniikki sovellustunnus, joten nimipalvelinta konfiguroitaessa ja SSL-sertifikaattia luotaessa tämä täytyy pitää mielessä. Sovelluksen sovellusalue, app domain, on sama jokaisessa farmin sovelluksessa.

## 3.4 SharePointin yleisimmät kielet ja kehitystyökalut

Sovelluskehitystä voidaan tehdä muun muassa HTML5:tä, JavaScriptiä, .NET:ä, PHP:tä sekä Javaa käyttäen. Sovelluskehitystä tehdään yleisimmiten käyttäen Microsoftin Visual Studio -ohjelmistoa erillisellä SharePoint-serverillä, johon otetaan yhteys jollakin Remote Desktopilla tai vaihtoehtoisesti kehitysympäristö voi olla virtuaalikoneella Azuressa. Visual Studio -kehitys vaatii Microsoftin tarjoaman O365 kehitystyökalu -lisäosan. Napa Office 365 -kehitystyökalu mahdollistaa sovelluskehityksen suoraan selaimessa, mutta siinä on huomattavasti vähemmän työkaluja, kuin Visual Studiossa. Napa asennetaan O365:ssa kehitysympäristöön sovelluksena. Office 365 -kehitysympäristö maksaa 99 $ vuodessa. Sen sijaan MSDN-tilausmaksut ovat huomattavasti kalliimpia ja voivat maksaa useita tuhansia euroja. MSDN-tilaukseen kuuluu riippuen tuotteesta muun mussa uusin Visual Studio, Azure, SharePoint, SQL Server, Office ja Exchange (MSDN 2015).

Sovelluksen suunnitteluvaiheessa valitaan jokin SharePointin tukema ohjelmointirajapinta, API. Valinta riippuu siitä, mitä sovelluksella halutaan tehdä, sillä rajapinnat soveltuvat eri käyttötarkoituksiin. Kuviosta 12 nähdään eri rajapintojen ominaisuudet. Esimerkiksi jos tavoitteena olisi luoda ASP.NET-

sovellus, joka hallinnoisi SharePointin dataa tai ulkopuolista dataa, voitaisiin käyttää .NET CSOM -kehystä. Yleisesti ottaen JavaScriptillä voidaan tehdä valtaosa tarvittavista asioista.




KUVIO 12. SharePoint 2013 -ohjelmointirajapinnat (MSDN 2014)

## 3.5 Farmi-ratkaisut ja NCSS

SharePoint 2010 -ympäristössä käytettiin ennen uutta sovellusmallia farmiratkaisuja, jotka asennettiin SharePoint-serverille ja jonka koodi ajettiin farmitasolla. Farmi-ratkaisut ovat SharePointin komponenttipaketteja, jotka tukevat farmitasolla ajettavaa custom hallittua koodia. Näitä ratkaisuita ei voi asentaa SharePoint Onlineen.

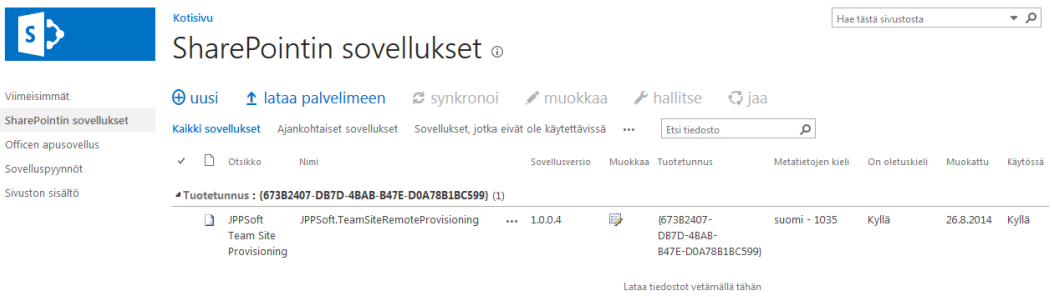
NCSS (no-code sandboxed solutions), eli kooditon hiekkalaatikkomalli on vanhempi ratkaisumalli. NCSS-ratkaisuita asennetaan paketteina suoraan SharePoint-komponenteiksi. Microsoft toivoo hiekkalaatikkomallin käytön välttämistä ja sen korvaamista sovellusmallilla mahdollisimman monessa tilanteessa (MSDN 2014). NCSS:llä voidaan silti tehdä joitakin asioita tehokkaammin kuin uudella sovellusmallilla. Jos halutaan ladata komponentteja SharePoint-ympäristöön ilman mitään erillistä vaadittua logiikkaa, voidaan se hoitaa NCSS-mallilla, jolloin erillistä sovellusverkkoa ei tarvitse ylläpitää

ainoastaan komponentteja varten. Hiekkalaatikkomalli toimii vielä toistaiseksi paikallisesti ja pilvessä. (MSDN 2014)

Farmi-ratkaisuiden ja NCSS-mallien käyttöä pyritään välttämään sovellusmallin tarjoamien etujen takia: Sovellusmallilla tehtyjä sovelluksia on helpompi levittää sovelluskaupan ja katalogin avulla, sovellusten turvallisuutta ja lupatasoja on helpompi säädellä sovellusverkon ansiosta, sovelluksia voidaan isännöidä muuallakin, kuin SharePoint-ympäristössä sekä sovellusmallilla voidaan hyötykäyttää HTML, REST, OData, JavaScript ja OAuth -standardeja (MSDN 2014).

## 3.6 Sovelluskatalogi ja kauppa

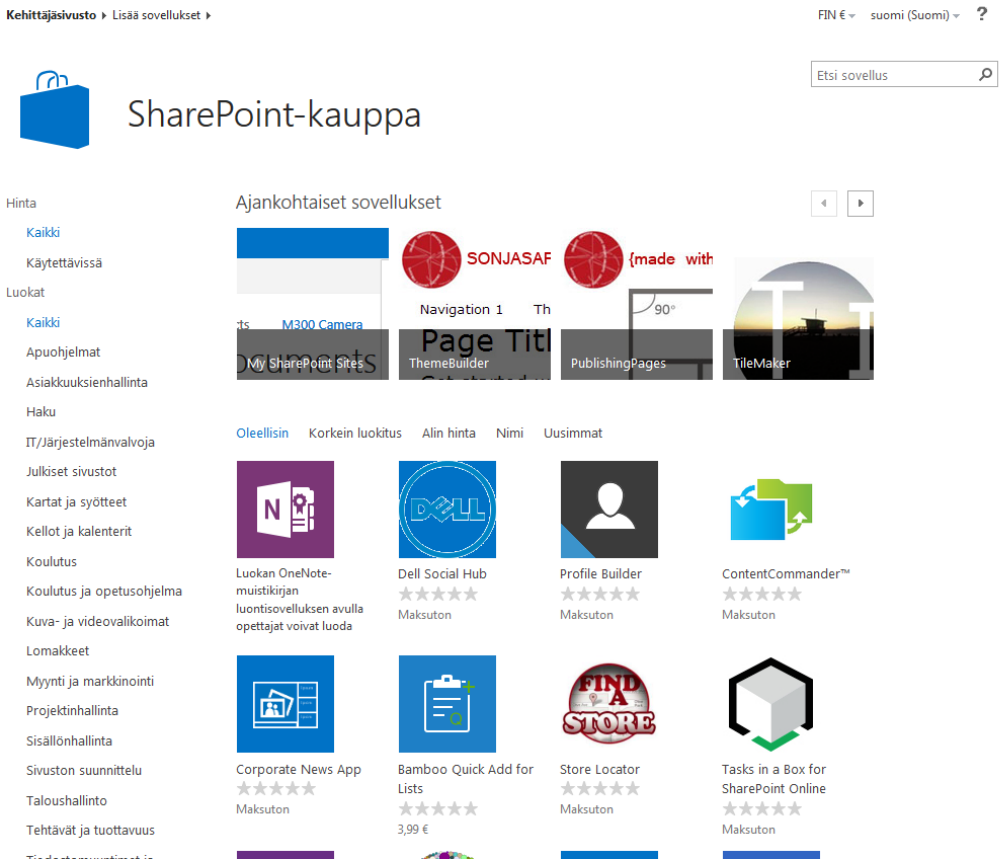
Sovelluksia voidaan ladata katalogiin, josta niitä voidaan jakaa tai asentaa suoraan eri sivustokokoelmiin. Sovelluskatalogi on erillinen sivustokokoelma SharePoint-farmissa. Katalogin kautta sovelluksia voidaan hallinnoida ja päivittää halutulla tavalla sekä niiden oikeustasoja voidaan muuttaa. Sovelluspäivityksiä voidaan ladata katalogiin, jolloin niitä tarjotaan sivustokokoelmissa, joissa kyseinen sovellus on käytössä. Katalogissa olevat sovellukset näkyvät sivustoilla omat sovellukset-välilehdellä. Kuviossa 13 katalogiin on lisätty yksi sovellus.




KUVIO 13. Sovelluskatalogi

SharePointin sovelluskauppa jakaa käyttäjien tekemiä sovelluksia ilmaiseksi tai maksua vastaan. Kuviossa 14 on sovelluskaupan etusivu. Maksulliset sovellukset käyvät Microsoftin tarkistusprosessin läpi, jossa niiden sisältö arvioidaan

sopivaksi. Tarkistusprosessissa varmistetaan muun muassa sovellusversioiden yhtenäisyys, sovellustunnuksen ainutkertaisuus, sovelluksen toimivuus ja OAuth-tunnusten yhtenäisyys (MSDN 2015).




KUVIO 14. Sovelluskauppa

# 4 SHAREPOINT SOVELLUSKEHITYS JA AZURE

Azurea voidaan käyttää muun muassa sovelluskehitysympäristönä sekä sovellusten isännöintipalveluna.

Kun Azuren hyödyntämistä pohdittiin yrityksessä ensimmäistä kertaa, tavoitteena oli sovellusten päivitysprosessin helpottaminen sekä custom-koodin siirtäminen pois SharePoint-ympäristöstä. Tämä perustuu siihen, että asiakkaiden sovellukset olisivat Azuressa, josta ne olisi helppo päivittää. Ilman Azurea, jokaisen sovellukset päivitys pitäisi suorittaa jokaisessa asiakasympäristössä erikseen, minkä takia päivitys veisi huomattavasti enemmän aikaa.

Sovelluksia ja etätapahtumankäsittelijöitä (remote event receiver, ReR) voidaan siirtää Azureen, josta niihin saadaan yhteys SharePointista (MSDN 2015). ReR:t ovat erillään SharePointista ajettavia palveluita, jotka toimivat muun muassa listojen ja lista-alkioiden tapahtumankäsittelijöinä.

Azuressa sijaitsevaan sovellukseen pääsee käsiksi käyttämällä uudelleenohjaussovellusta SharePointissa. Udelleenohjaussovellus linkittää Azuressa isännöitävän sovelluksen SharePointtiin, ja saa sille myönnetyt oikeudet SharePoint-palvelimella.

## 4.1 Sovelluskehitysympäristö Azuressa

Azureen voidaan luoda virtuaalikoneella toimiva sovelluskehitysympäristö MSDN-tunnusten avulla (MSDN 2015). Kehitysympäristö luodaan valmiista levykuvasta, ja siinä on valmiiksi asennettuna tarvittavat kehitystyökalut, kuten uusin Visual Studio ja SharePoint 2013. Valmis levykuva kuuluu Azuren tuotteisiin, joten sellaista ei tarvitse luoda erikseen. Levykuva provisioidaan selaimesta Azure-hallintaportaali-sivuilta ja sen valmistuttua kehitysympäristöön voidaan ottaa etäyhteys Remote Desktop -työkalulla.

## 4.2 Azuren valmistelu sovelluksen isännöintiympäristöksi

Jotta Azurea voitaisiin käyttää sovelluksen isännöintiympäristönä, täytyy sinne luoda oma verkkosivusto sovelluksia varten. Azure-sivusto näyttää yleensä seuraavalta: ”https://azuresivusto.azurewebsites.net”. Sivustolle julkaistaan esimerkiksi Visual Studion avulla isännöitävä sovellus. SharePoint-sovelluksen julkaiseminen Azuressa vaatii julkaisuprofiilin, joka voidaan luoda Azure-verkkosivulla.

MSDN-tilauksella voidaan saada kymmenen Azure-verkkosivua ilmaiseksi. Yhtä verkkosivua voidaan käyttää isännöimään useita sovelluksia. Jotta tämä olisi mahdollista, täytyy verkkosivusto avata FTP-yhteydellä ja lisätä hakemistoon jokaista sovellusta kohden oma virtuaalihakemisto. Virtuaalihakemistot täytyy rekisteröidä ja ottaa käyttöön verkkosivuston hallintaportaalissa.

## 4.3 SharePoint sovelluksen rekisteröiminen Azureen

Jotta sovellukseen pääsisi SharePoint-ympäristöstä ja jotta Azuressa sijaitseva sovellus voisi käyttää SharePointtia, täytyy sovellukselle rekisteröidä asiakasohjelman tunnukset SharePointissa. Yhteyden luomiseksi tarvitaan SharePointin puolelta client ID, joka on sovelluksen GUID-koodi, sovelluksen salasana eli client secret, sovelluksen näyttönimi sekä Azuren URL-osoite. Client ID ja Client secret voidaan generoida SharePointin puolelta käyttämällä AppRegNew.aspx-sovellusta. Sovellus käynnistetään selaimessa kirjoittamalla SharePoint-osoitteen perään ”/_layouts/15/AppRegNew.aspx” (MSDN 2014). Generoidut koodit tulee kirjata sovelluksen AppManifest.xml-tiedostoon sekä web.config-tiedostoon ennen sovelluksen julkaisua Azureen. Kuviossa 15 on sovellusrekisteröinnin lomake; asiakasohjelman tunnus sekä asiakasohjelman salasana -kenttien vieressä on ”Luo”-painike, jota painamalla tarvittavat tunnukset generoidaan.


KUVIO 15. Sovelluksen rekisteröintilomake eli AppRegNew.aspx

## 4.4 Sovelluksen virheiden testaus Azuressa

Etäkoodin testaus onnistuu Azuressa Service Bus -työkalun avulla. Palvelulinja luodaan tekemällä ensin linjaa varten oma nimiavaruus Azuressa. Azure tarjoaa Shared Access Signature- (SAS) sekä Access Control Service (ACS) -tekniikat linjan todentamista varten, mutta tällä hetkellä ainoastaan ACS-todennusta voidaan käyttää etäkoodin testaukseen.

ACS-merkkijonon saa käyttöönsä, kun Azure-nimiavaruus luodaan Windows Azure PowerShelliä käyttäen. Nimiavaruutta luotaessa luontilauseeseen tulee lisätä -CreateACSNamespace $true lauseen perään. Jos nimiavaruus luodaan Azuren hallintaportaalin kautta, ei ACS-merkkijonoa saada käyttöön syyskuun 2014 jälkeen luoduissa nimiavaruuksissa (MSDN 2015). Kuviossa 16 nähdään esimerkkilause nimiavaruuden luonnista Azure Powershellillä.

```
PS C:\> New-AzureSBNamespace "esimerkkiavaruus" "North-Europe" -CreateACSNamespace $true -NamespaceType Messaging
```

KUVIO 16. Nimiavaruuden luonti

## 4.5 Azuren käyttö

Azuren käyttö sovelluskehityksessä on lähes ilmaista, sillä se kuuluu MSDN-tilaukseen. Riippuen tilaustasosta, Azuren palveluita voidaan käyttää tiettyyn käyttömäärään asti. Sovellusten asentaminen Azureen on helppoa, ja sen

käyttäminen ohittaa monta arkkitehtuurillista ongelmaa, joiden suunnittelemiseen ja toteuttamiseen olisi kulunut aikaa ja rahaa muissa pilvipalveluissa. Paikallisten palvelimien ylläpitäminen sovelluskehitystä ja sovellusten isännöintiä varten vähenee Azuren käytön ansiosta. Muina etuina saadaan hyvä varmuuskopiointi sekä korkea käynnissäoloaika ja skaalautuvuus. SharePoint tukee 2013-versiosta lähtien Azurea luontaisesti (MSDN 2014).

# 5 REMOTE PROVISIONING APP

Opinnäytetyönä toteutettu sovellus on osa laajempaa tuotepakettia, jota myydään lisäpalveluna muissa SharePoint-ratkaisuissa. Kokonaista tuotetta tarjotaan suuren- ja keskitason yrityksille auttamaan projektinhallinnassa. Kaikissa suunnitelluissa asiakastapauksissa luotujen työtilojen on tarkoitus sijaita pilvessä O365-ympäristössä, vaikka muu asiakastoteutus sijaitsisikin paikallisesti SharePoint-palvelimella. Työtilojen toteutukset ovat siis joko täysin pilvessä toimivia ympäristöjä tai hybridi-toteutuksia.

Toteutettu ohjelma on edellämainittuun tuotepakettiin kuuluva sovellus, jonka tarkoituksena on luoda customoituja työtila-sivustoja tiettyjen kategoria-sivustojen alisivustoiksi. Sovelluksen logiikka isännöidään Azuressa, josta työtilojen luonti tapahtuu SharePoint-ympäristöön. SharePointissa sijaitsee uudelleenohjaus-sovellus, jolle myönnetään tarvittavat oikeudet ja joka avaa käynnistettäessä varsinaisen Azuressa sijaitsevan sovelluksen.

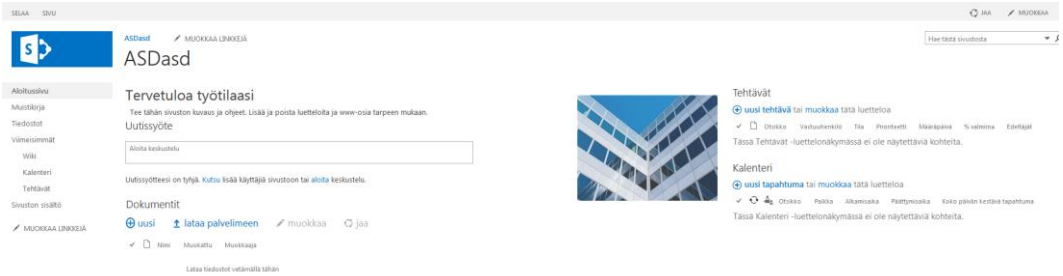
Ohjelmaa käyttämällä halutaan siirtää custom-koodia pois SharePoint -ympäristöstä, sekä helpottaa sovellusten hallintaa siirtämällä mahdollisimman monien asiakkaiden sovelluksia yhteen paikkaan. Sovellukseen kuuluu SharePoint-sovelluksille tyypillinen aspx.sivu, joka toimii sovelluksen etusivuna ja lomakkeena. Aspx-sivu käyttää JavaScriptiä ja CSS-tiedostoja sivun tyylittämiseen sekä linkittämään SharePointin ylätunnisteet sovellukseen. Ohjelman sivustonluontilogiikka on kirjoitettu C#-kielellä SharePoint 2013 .NET-kehyksen CSOM-mallilla.

## 5.1 Työtila

Työtila on wikipage-pohjainen dokumenttienjako-sivusto. Ideana on, että jokaisella projektilla olisi oma työtila-sivusto, jossa voidaan jakaa projektille ominaisia asiakirjoja, keskustella projektista sekä seurata sen kulkua. Työtila-sivuja luodaan kategoria-sivujen alisivuiksi. Kategoriat eivät itsessään ole työtiloja, vaan ne rajaavat varsinaiset työtilat luokkiin, kuten markkinointi, myynti ja projektit.

Työtiloja isännöidään SharePoint online -versiossa pilvessä, vaikka itse intranet olisikin vain lähiverkossa sijaitseva sivusto. Tässä hybridimallissa on se etuna, että dokumentteja sekä työtiloja voidaan tarvittaessa jakaa organisaation ulkopuolisille tahoille ja dokumentteihin pääsisi käsiksi myös intranetin ulkopuolelta.

Sivustoista on useita eri customoituja malleja, jotka valitaan työtilaa luotaessa sovelluksesta. Kuviossa 17 on esimerkki luodusta custom-työtilasta. Kuviossa on neljä verkko-osaa, joista kolme vimeisintä on yhdistetty samannimisiin listoihin muokatulla näkymällä.




KUVIO 17. Esimerkki luodusta työtilasta

## 5.2 Remote Provisiointi

Työtila-sivustoja luodaan työtila-portaalin alisivustoina olevien kategorioiden alisivustoiksi. CSOM-mallissa sivustoja voidaan luoda SharePoint.Client-nimiavaruudessa olevalla ClientContext-objektilla. ClientContextille syötetään sivuston URL-osoite, jolloin sen konteksti saadaan käyttöön. Provisiointisovellus käynnistetään työtila-portaalin kautta, joten erääksi ongelmaksi nousi, miten työtiloja voitaisiin luoda kategoria-sivujen alle, kun kontekstiin on ladattu ainoastaan työtila-portaali. Ongelma ratkaistiin etsimällä käytössä olevasta ClientContextista olevat alisivut ja vertaamalla niiden nimiä lomakkeesta valittuun kategoriaan. Nimen perusteella pystyttiin muodostamaan kategoria-sivun URL-osoite, jonka avulla kategoria-sivun ClientContext saatiin käyttöön.

Sivuston luontiprosessi meni vaihevaiheelta näin:

1. Avataan SharePointissa sijaitseva uudelleenohjaus-sovellus, joka yhdistää provider-hosted-sovellukseen Azuressa.
2. Azuressa sijaitseva sovellus tarjoaa erilaisia pohjia vaihtoehdoiksi sivuston luomiseen sekä kysyy työtilan kategorian ja halutun URL-osoitteen.
3. Etsitään kategoria-sivuston konteksti käyttöön sivuston luontia varten.
4. Sivusto luodaan CSOM:n avulla SharePointtiin valitulla pohjalla ja oikeuksien perintä katkaistaan.
5. Ominaisuudet ja toiminnot aktivoidaan provisioidulla sivustolla.

## 5.3 Uudelleenohjaus-sovellus

SharePointtiin ladataan uudelleenohjaus-sovellus linkittämään Azureen julkaistu sovellus SharePointtiin. Sovelluksessa ei tarvitse olla muuta, kuin AppManifest.xml-tiedosto, jossa on vähintään seuraavat tiedot: Azuressa sijaitsevan sovelluksen osoite, yhtenäinen generoitu clientID sekä tarvittavien oikeuksien pyyntö. Alisivustojen luonti vaatii täydet oikeudet sivustokokoelmassa, joten ne myönnetään tässä tapauksessa uudelleenohjaus-sovellukselle. Kuviosta 18 nähdään, miten oikeuksien pyyntö ja sovellusten linkitys tapahtuu AppManifestissa. Tarvittavien metatietojen, kuten sovelluksen näyttönimi ja versiointi, olisi hyvä olla kyseisessä tiedostossa.

```
<App xmlns="http://schemas.microsoft.com/sharepoint/2012/app/manifest"
     Name="JPPSoftTeamSiteProvisioning"
     ProductID="{673b2407-db7d-4bab-b47e-d0a78b1bc599}"
     Version="1.0.0.4"
     SharePointMinVersion="15.0.0.0"
>
  <Properties>
    <Title>JPPSoft Team Site Provisioning</Title>
    <StartPage>https://istodev.azurewebsites.net/Pages/Default.aspx?{StandardTokens}</StartPage>
  </Properties>

  <AppPrincipal>
    <RemoteWebApplication ClientId="a6b80ad8[redacted]505f2e" />
  </AppPrincipal>
  <AppPermissionRequests AllowAppOnlyPolicy="true">
    <AppPermissionRequest Scope="http://sharepoint/content/sitecollection" Right="FullControl" />
    <AppPermissionRequest Scope="http://sharepoint/content/sitecollection/web" Right="FullControl" />
  </AppPermissionRequests>
</App>
```

KUVIO 18. AppManifest.xml

Kehitysvaiheessa uudelleenohjaus-sovellus voitiin ladata kehitysympäristöön eikä sitä tarvinnut päivättää. Kaikki päivitettävät osat sijaitsivat varsinaisessa sovelluksessa Azuressa, joten jos sovelluksen valmisversio saisi ikinä päivityksiä, ei SharePointin asiakasympäristössä tarvitsisi päivittää mitään ohjelmaan liittyen.

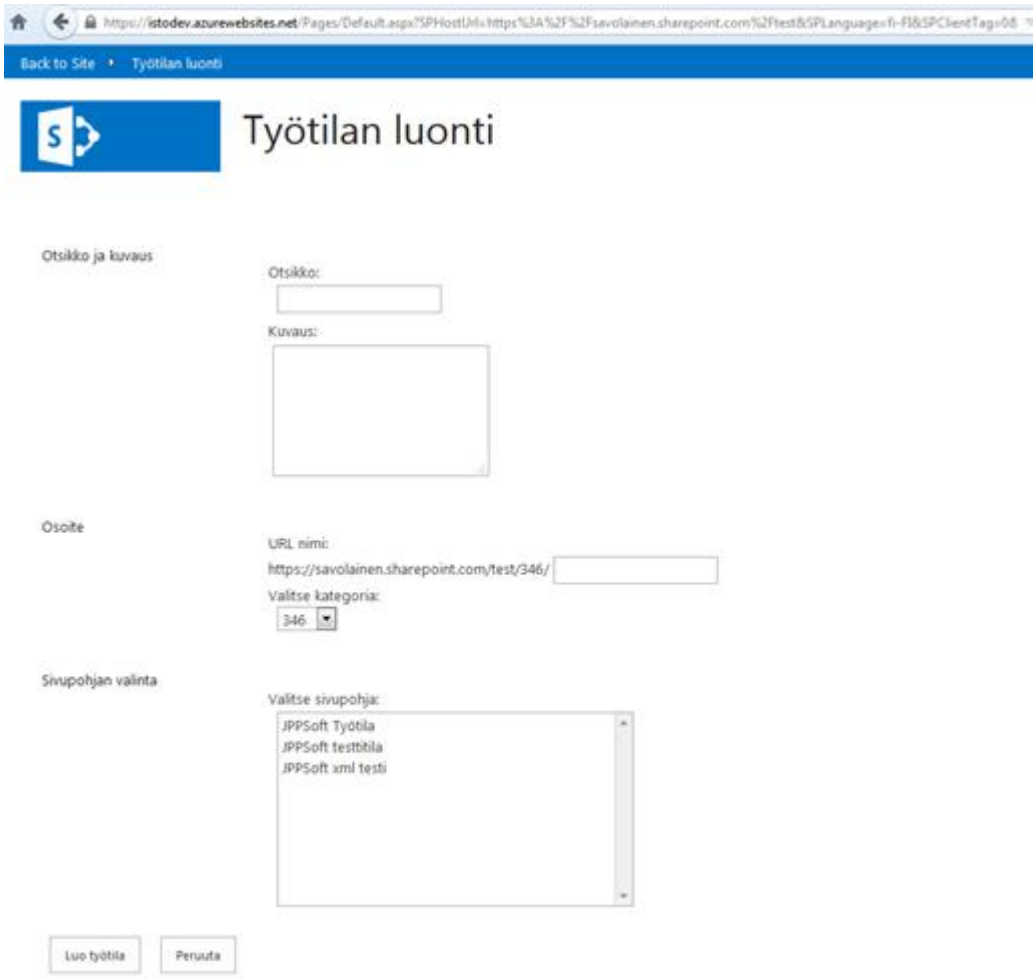
## 5.4 Käyttäkäkokemus

Sivuston luominen aloitettiin painamalla sille luotua painiketta Työtilaportaalin etusivulta. Sivupohjaan on kiinnitetty painike, joka on ympyröity Security Trim -katkelmalla. Security Trim -katkelman tarkoitus on rajoittaa halutun elementin näkyvyyttä käyttöoikeuksien mukaan (MSDN 2013). Tässä tapauksessa vain käyttäjät, joilla on sivustonluontioikeus, voivat nähdä painikkeen, jolla sovellus käynnistetään.

Siirtyminen Azuressa sijaitsevaan sovellukseen, ja ylipäätänsä SharePoint-sivulta sovellukseen, pyrittiin toteuttamaan mahdollisimman huomaamattomasti ja noudattamaan SharePointin senhetkistä Master-tyylitiedostoa. Sivustonluonti pyrittiin toteuttamaan siten, että käyttäjä pitäisi sovellusta SharePointin natiivina vakio-ominaisuutena eikä huomaisi siirtymistä SharePointista pilveen. Tämä pystyttiin saavuttamaan käyttämällä SharePointin vakioita tyylejä. Painikekuvia on matkittu ja käytetty hyväksi sovelluksen ulkonäössä. Chrome Control -ominaisuus aktivoitiin sovelluksen lomakesivulla SharePointin omien tyylitiedostojen sekä ylätunnisteen käyttöön ottamiseksi.

SharePointille natiivi sivunluomisprosessi pystyttiin korvaamaan sovelluksella injektoimalla sovelluksen käynnistys natiiviin sivustonluonti-painikkeeseen, mutta kehitysvaiheessa ominaisuutta ei todettu tarpeelliseksi SharePointin natiivin sivustonluontiominaisuuden säilyttämiseksi. Kuviossa 19 nähdään sovelluksen sivustonluontilomake. Sivun ylälaidassa on painike Back to Site, jolla voidaan palata työtilojen etusivulle. Painike on sivulla Chrome control -ominaisuuden ansiosta. Sivun URL-osoitteesta nähdään, että sovellus sijaitsee Azure-verkkosivulla, mutta varsinainen lomake ja latausruudut antavat kuvan, että sovellus sijaitsisi SharePoint-ympäristössä. Lomakkeessa voidaan valita osoite ja

kategoria, jonka alle luotava työtila provisioidaan. Tässä tapauksessa kategoriaksi on valittu 346.




KUVIO 19. Sivustonluontilomake

Työtilaa luotaessa siitä katkaistaan oikeuksienperintä yläsivuilta ja sivuston luojalle myönnetään ylläpitäjän oikeudet työtilaan. Oikeuksienperinnän katkaisemisesta on se hyöty, että työtiloille voidaan asettaa omia ylläpitäjiä, jotka olisivat osa työtilan käyttäjistä. Samalla mahdollistetaan ulkopuolisten tahojen lisääminen tiettyjen työtilojen käyttäjiksi antamatta heille oikeuksia muihin yksityisempiin työtiloihin tai alisivustoihin. Jos oikeudet peritään yläsivuilta, pitäisi näille käyttäjille myöntää ylläpitäjän oikeudet myös yläsivustoille.

## 5.5 Työtilan luonti sivupohjista

Työtila-sivustojen luomiseen käytetään koodissa Microsoft.SharePoint.Client-kirjastoa. Kun työtilaa luodaan sovelluksella, sille valitaan jokin sivupohja. Sivupohjat on kuvailtu XML-tiedostossa. Jos sivupohjia haluttaisiin tehdä jälkikäteen lisää valmiiseen sovellukseen, onnistuu se muokkaamalla kyseistä XML-tiedostoa. Itse sovelluksen lähdekoodiin ei tarvitse tehdä muutoksia. Template.xml-tiedostossa voidaan määritellä jokaiselle sivupohjalle

- pohjan nimi
- kuvaus
- listat
- alisivut
- aloitus-sivu
- navigaation linkit
- web-osat
- näkymät.

Määritellyille listoille annetaan nimi, kuvaus, kotikansio ja listatyyppi. Jokainen SharePointin lista perustuu johonkin listatyyppiin, jota käytetään pohjana määriteltäessä listan kentät. Kuviossa 20 kuvaillaan Tehtävät-niminen lista, joka perustuu SharePointin listatyyppiin Tasks.

```
<Lists>
  <ListInstance
    Description = "Tehtävät" DocumentTemplate = ""
    OnQuickLaunch = "true" QuickLaunchUrl = "/_layouts/15/start.aspx#/Tehtvt/Forms/AllItems.aspx"
    TemplateType = "107" Title = "Tehtävät" Url = "Lists/Tehtävät"/>
</Lists>
```

KUVIO 20. Listan kuvaus tiedostossa

Näkymälle voidaan määritellä nimi, näyttönimi, rivimäärä, sivutus, näkymän kysely, kentät ja kenttien nimet. Kuviossa 21 kuvaillaan Tiedostot-listaan lisättävä näkymä Viimeksimuokattu. Näkymässä on kuusi riviä, eikä sitä ole sivutettu. Listan kysely ja kenttien nimet on muodostettu ensin luomalla vastaava lista

SharePointin listanluonti-työkalulla. Lista on tämän jälkeen avattu SharePoint Design Manager -työkalulla, josta nähdään kenttien nimet ja kysely.

```
<Views>
  <View List="Tiedostot" Title="Viimeksimuokattu" Rows="6" Paged="false" >
    <Query>
      <![CDATA[<OrderBy><FieldRef Name="Modified" Ascending="FALSE" /></OrderBy>]]>
    </Query>
    <Fields>
      <Field>DocIcon</Field>
      <Field>LinkFilename</Field>
      <Field>Modified</Field>
    </Fields>
  </View>
</Views>
```

KUVIO 21. Näkymän kuvaus tiedostossa

XML-tiedostossa voidaan käyttää joko custom-verkko-osia tai SharePointtiin valmiiksi asennettuja verkko-osia. Jos käytetään sivustolla jo sijaitsevaa verkko-osaa, ainoastaan sen nimi annetaan sivupohjassa. Jos halutaan käyttöön customoitu verkko-osa, annetaan sille jokin nimi, ja liitetään mukaan verkko-osan XML-skeema. Esimerkki verkko-osan XML-skeemasta on kuviossa 22. Verkko-osien kuvaus on XML-koodia, jossa kerrotaan kaikki tarvittavat komponentit, kuten listan näkymä ja osoite, jota verkko-osa käyttää.

Verkko-osat liitetään työtilan luonnissa sivulle käyttämällä sivun lähdekoodissa sijaitsevia placeholder-arvoja, tokeneita. Lähdekoodiin upotetut [{0}]-tokenit korvataan sivunluonnin yhteydessä verkko-osan koodilla siinä järjestyksessä, missä verkko-osat on määritelty XML-tiedostossa. Jos verkko-osa käyttää jotakin listaa, liitetään listan nimi ja listan näkymän nimi verkko-osan lähdekoodiin dynaamisesti korvaamalla upotetut tokenit XML:n lukuvaiheessa. Kuviossa 22 esitellään erään verkko-osan XML-skeema, joka liitetään cdata-muodossa sivupohja XML-tiedostoon. Riveillä 13 ja 14 kuvataan verkko-osassa käytettävän listan nimi ja GUID-koodi. Arvot ovat edellämainittuja upotettuja tokeneita, jotka korvataan oikeilla arvoilla dynaamisesti siinä vaiheessa, kun verkko-osa liitetään sivupohjaan.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <webParts>
3    <webPart xmlns="http://schemas.microsoft.com/WebPart/v3">
4      <metaData>
5        <type name="Microsoft.SharePoint.WebPartPages.XsltListViewWebPart,
6          Microsoft.SharePoint,
7          Version=15.0.0.0,
8          Culture=neutral,
9          PublicKeyToken=71e9bce111e9429c" />
10     </metaData>
11     <data>
12       <properties>
13         <property name="ListName" type="string">[{0}]</property>
14         <property name="ViewGuid" type="string">[{1}]</property>
15       </properties>
16     </data>
17   </webPart>
18 </webParts>
```

KUVIO 22. Verkko-osan skeema

Sivuja luotaessa käytetään joko XML-tiedostoon liitettyä sivun lähdekoodia, tai jossakin tiedostossa sijaitsevaa lähdekoodia, jos tiedoston nimi on annettu Template-määreellä. Kuviossa 23 esitellään eräs XML-tiedostoon liitetty wiki-tyyppisen sivupohjan HTML-lähdekoodi. Sivulla on kolme palstaa ja ylätunniste sekä yksi verkko-osa. Sivuston palstojen määrä sekä ylä- ja alatunnisteen käyttö määritellään riveillä 35 - 37, jossa rivillä 36 ensimmäinen arvo määrittelee ylätunnisteen käytön, toinen arvo alatunnisteen käytön ja kolmas arvo palstojen määrän. Verkko-osan XML-skeema liitetään rivillä 23 olevan [{0}]-tokenin tilalle sivustonluonnin yhteydessä. Esitelty HTML-kuvaus käännetään SharePointin puolella sisäisesti perinteisemmän näköiseksi HTML-koodiksi.

```
<div class="externalclassd1a150d6187f449b8a6c4bea2d4913bb">
    <table id="layoutstable"style="width:100%;">
        <tbody>
            <tr style="vertical-align:top;">
                <td colspan="3">
                    <div class="ms-rte-layoutszone-outer"style="width:100%;">
                        <div class="ms-rte-layoutszone-inner"role="textbox"aria-haspopup="true"aria-autocomplete="both"aria-multiline="true">
                        </div>
                    </div>
                </td>
            </tr>
            <tr style="vertical-align:top;">
                <td style="width:33.3%;">
                    <div class="ms-rte-layoutszone-outer"style="width:100%;">
                        <div class="ms-rte-layoutszone-inner"role="textbox"aria-haspopup="true"aria-autocomplete="both"aria-multiline="true">
                        </div>
                    </div>
                </td>
                <td class="ms-wiki-columnspacing"style="width:33.3%;">
                    <div class="ms-rte-layoutszone-outer"style="width:100%;">
                        <div class="ms-rte-layoutszone-inner"role="textbox"aria-haspopup="true"aria-autocomplete="both"aria-multiline="true">
                            [{0}]
                        </div>
                    </div>
                </td>
                <td class="ms-wiki-columnspacing"style="width:33.3%;">
                    <div class="ms-rte-layoutszone-outer"style="width:100%;">
                        <div class="ms-rte-layoutszone-inner"role="textbox"aria-haspopup="true"aria-autocomplete="both"aria-multiline="true">
                        </div>
                    </div>
                </td>
            </tr>
        </tbody>
    </table>
    <span id="layoutsdata"style="display:none;">
        true,false,3
    </span>
</div>
```

KUVIO 23. Wiki-tyyppisen sivun HTML-pohja

## 5.6 Työkalut

Projektissa työkaluina käytettiin henkilökohtaista O365-kehitysympäristöä koodin testaukseen, Azure-verkkosivua koodin isännöitiin, Visual Studio 2013, O365 -kehityslisäosalla koodin editointiin ja julkaisemiseen sekä Team Foundation Serveriä versiohallintaan. Projektista tehtiin asiakaskohtaiset versiot, jotta tarvittavat brändäykset ja muut asiakaskohtaiset ominaisuudet pysyisivät erillään. Kehitystyötä helpotti Visual Studion ominaisuus julkaista sovellus suoraan O365:een ja Azureen. Koodin virheentestaus, eli debuggaus, hoidettiin Visual Studiolla. Julkaistu koodi sijaitsi Azuressa, joten virheentestauksen toimintaan saaminen vaati Service Bus -ominaisuuden käyttöä, jonka toimintaan saamisesta kerrottiin aikaisemmissa luvuissa.

Sovelluksen testausta varten suunniteltiin useita testitapauksia ajettavaksi säännöllisin väliajoin. Testitapaukset olivat manuaalisia vaihe vaiheelta tehtäviä testejä, joiden suunnitelmat ja tulokset kirjattiin selaimessa toimivaan TestRail-ohjelmaan. Varsinaisia automatisoituja testejä eikä yksikkötestejä ollut vielä toteutettu. TestRailin tarjoamilla työkaluilla pystyttiin suunnittelemaan yksityiskohtaisia testejä, joiden avulla projektin ulkopuolinen voisi tehdä

määritellyt testit ja kirjata tulokset ylös. Testitapaukset kuvattiin vaihe vaiheelta ja jokaisessa vaiheessa oli selkeästi merkattu toivottu tulos kuvakaappauksineen.

## 5.7 Kehityshaasteet

Microsoftilla on tunnetusti laaja dokumentaatio sekä SharePointista, Azuresta että .NET-ympäristöistä, mitä voitiin hyödyntää projektin aikana tehokkaasti. Aiheesta on runsaasti ammattilaisten kirjoituksia useissa SharePoint-aiheisissa blogeissa sekä Microsoftilta että muilta alan ammattilaisilta.

Projektin aikana oli lukuisia haasteita, kuten virheentestauksen toimiminen sekä oman Azure-ympäristön puuttuminen, mutta henkilökohtaisesti isoimmaksi haasteeksi nousi lähinnä ajan puute. Sovellusta alettiin kehittämään työsuhteen loppuvaiheilla, jonka takia sovelluksesta saatiin ainoastaan toimiva ensimmäinen versio valmiiksi. Kehitystyötä olisi voinut vielä jatkaa, ja joitain pienempiä ominaisuuksia jäi tekemättä. Sovelluksen ensimmäinen versio kuitenkin asennettiin muutamiin asiakasympäristöihin käyttöön, ja sovellus jäi vielä jatkokehitykseen työsuhteen loppumisen jälkeen.

## 5.8 Jatkokehitys

Asennuksen jälkeen templateja voidaan muokata xml-tiedostossa. Jatkokehitystä miettien xml-tiedostolle voisi kehitellä editorin, jolla tiedoston muokkaaminen olisi helpompaa. Yksikkötestejä sekä asennuksen jälkeisiä automatisoituja käyttöliittymätestejä olisi mahdollista tehdä sovellukselle. Automatisoidut testit auttaisivat sovelluksen toiminnan varmistamisessa sekä jatkokehityksessä että jo valmiiksi asennetussa ympäristössä.

Sovelluksella pystyy tällä hetkellä luomaan vain wiki-pohjaisia sivuja sivustoille ja ainoastaan yhden kappaleen, joka toimii sivuston kotisivuna. Sivustonluontipohjiin voitaisiin toteuttaa tuki monen sivun provisioimiseen samalla kertaa sekä julkaisupohjaisten sivujen luomisen.

Tällä hetkellä sivustojen luonnin voi aloittaa ainoastaan käyttäjät, joilla on sivustonluontioikeudet kokoelmassa. Sovellukseen voitaisiin kehittää järjestelmä,

jotta vähemmän oikeuksia pitävät käyttäjät voisivat pyytää työtilasivustojen luontia.

Provider-hosted-sovellusmallia voitaisiin hyötykäyttää muissakin tulevissa ja edellisissä sovelluksissa. Vanhojen sovelluksien muuntaminen esiteltyyn muotoon ei välttämättä olisi liian aikaavievä projekti.

# 6 YHTEENVETO

SharePoint-sovelluskehitys on erittäin laaja kokonaisuus ja silti vain pieni osa koko SharePoint-kehityksestä. Jo valmiiksi useita käytettyjä ohjelmointikieliä osaava ohjelmoija löytää haasteita jatkuvasti muuttuvasta SharePoint-ympäristöstä.

Provider-hosted apps -malli sekä Azure olivat opinnäytetyön alussa henkilökohtaisesti uusia asioita ja vaativatkin enemmän opiskelua, vaikka SharePoint-sovelluskehityksestä olikin jo entuudestaan kokemusta. Sovelluskehityskielenä oli aikaisemmin käytetty JavaScriptiä ja REST-mallia, mutta projektiin otettiin silti käyttöön C#-kieli ja CSOM, jotka erosivat aikaisemmin opitusta useammallakin tavalla. Kokeneemmat SharePoint-kehittäjät olivat projektin aikana suureksi avuksi, ja ongelmien ratkaisuun sekä perehtymiseen kului sen takia huomattavasti vähemmän aikaa.

Toteutetun sovelluksen avulla pyrittiin vähentämään custom-koodin määrää SharePoint-ympäristöstä siirtämällä joitakin sovelluksia Azureen. Sovelluksesta saatiin tehtyä ensimmäinen toimiva versio, joka otettiin käyttöön muutamissa asiakasympäristöissä. Samalla todettiin Azuressa isännöitävät sovellukset käyttökelpoiseksi ideaksi. Custom-koodi ei ole kokonaan kadonnut asiakasympäristöistä, mutta osittain sovelluksen ansiosta se on ainakin vähentynyt. Sovellusratkaisu todettiin toimivaksi, ja se jäi vielä jatkokehitykseen, joten sovelluksen kokonaishyötyä syksyllä 2014 ei päästy näkemään. Provider-hosted-sovellusmallia käytetään vielä tulevaisuudenkin sovelluskehitysratkaisuissa.

Suurimpia haasteita projektissa oli tuotetun koodin virhetestaus etänä Azuressa, jonka toimiminen oli sovelluskehityksen kannalta lähes välttämätöntä. Virheentestaus saatiin toimintaan, mutta siihen uhrattiin useita työtunteja. Käytännön osuuden valmistuttua SharePoint sovelluskehityksen ja Azuren tietämys on noussut huomattavasti lähtötasoon nähden, ja toivottavasti SharePointin tuntemuksesta olisi hyötyä tulevaisuudessakin.

LÄHTEET

Microsoft. 2014. Introduction to Microsoft Azure [viitattu 14.3.2015]. Saatavissa: http://azure.microsoft.com/en-us/documentation/articles/fundamentals-introduction-to-azure/

Microsoft Office Support. 2015. Create and use site templates [viitattu 30.2.2015]. Saatavissa: https://support.office.com/en-au/article/Create-and-use-site-templates-60371b0f-00e0-4c49-a844-34759ebdd989

MSDN. 2011. Introduction to CAML. [viitattu 30.3.2015]. Saatavissa: https://msdn.microsoft.com/en-us/library/office/ms426449.aspx

MSDN. 2013-2015. Microsoft Development Network. SharePoint-dokumentaatio [viitattu 1.3.2015]. Saatavissa: https://msdn.microsoft.com/en-us/library/office/fp161507.aspx

MSDN. 2013 a. How to: Add a Security Trim snippet in SharePoint 2013. [viitattu 25.3.2015]. Saatavissa: https://msdn.microsoft.com/en-us/library/office/jj822366.aspx

MSDN. 2013 b. Overview of the SharePoint 2013 page model. [viitattu 30.1.2015]. Saatavissa: https://msdn.microsoft.com/en-us/library/office/jj191506.aspx

MSDN. 2014 a. Guidelines for registering apps for SharePoint 2013 [viitattu 30.2.2015]. Saatavissa: https://msdn.microsoft.com/en-us/library/office/jj687469.aspx

MSDN. 2014 b. Apsp for SharePoint compared with SharePoint solutions [viitattu 18.2.2015]. Saatavissa: https://msdn.microsoft.com/en-us/library/office/jj163114.aspx

MSDN. 2015 a. Visual Studio. MSDN Subscriptions [viitattu 15.3.2015]. Saatavissa: https://www.visualstudio.com/en-us/products/msdn-subscriptions-vs.aspx

MSDN. 2015 b. Get started with SharePoint 2013 REST service. [viitattu 30.3.2015]. Saatavissa: https://msdn.microsoft.com/EN-US/library/office/fp142380.aspx

MSDN. 2015 c. Service Bus Authentication and Authorization [viitattu 13.3.2015]. Saatavissa: https://msdn.microsoft.com/en-us/library/azure/dn170478.aspx

MSDN. 2015 d. Validation policies for apps FAQ. [viitattu 20.3.2015]. Saatavissa: https://msdn.microsoft.com/en-us/library/office/jj591603.aspx

MSDN. 2015 e. Choose the right API set in SharePoint 2013. [viitattu 29.4.2015]. Saatavissa: https://msdn.microsoft.com/en-us/library/office/jj164060.aspx

MSDN. 2015 f. Host apps for SharePoint 2013 on a Microsoft Azure Cloud Service. [viitattu 29.4.2015]. Saatavissa: https://msdn.microsoft.com/en-us/library/office/dn343301.aspx

Office. 2015. Create and use site templates. [viitattu 29.4.2015]. Saatavissa: https://support.office.com/en-au/article/Create-and-use-site-templates-60371b0f-00e0-4c49-a844-34759ebdd989

SharePoint developer blog. 2015. Deprecation of Custom Code in Sandboxed Solutions [viitattu 30.1.2015]. Saatavissa: http://blogs.msdn.com/b/sharepointdev/archive/2014/01/14/deprecation-of-custom-code-in-sandboxed-solutions.aspx

Technet. 2014 a. Overview of hybrid SharePoint 2013 for technical decision makers [viitattu 15.1.2015]. Saatavissa: https://technet.microsoft.com/en-us/library/dn607308.aspx

Technet. 2014 b. Microsoft Architectures for SharePoint 2013. [viitattu 25.3.2015]. Saatavissa: https://technet.microsoft.com/en-us/library/dn635309.aspx

Technet. 2014 c. Overview of sites and site collections in SharePoint 2013. [viitattu 29.4.2015]. Saatavissa: https://technet.microsoft.com/en-us/library/cc262410.aspx

Whitehead. 2013. How to configure SharePoint 2013 On-Premises Deployment for Apps [viitattu 1.3.2015]. Saatavissa: http://blogs.technet.com/b/mspfe/archive/2013/01/31/configuring-sharepoint-on-premise-deployments-for-apps.aspx